COBB®
Inside
MICROSOFT ACCESS™
August 1993 • Vol. 1 No. 6
Tips & techniques for Microsoft Access • Windows

Report Tip

Creating delivery-route reports by sorting even and odd street numbers

Have you ever needed to create a report that sorts address data? For invoices or shipping reports, you usually don't need to organize the data by addresses. However, if you sort addresses for people who sell or deliver goods door to door, you'll make the delivery job much easier if you group the data by even and odd street numbers and by block. In this article, we'll show you how to use Access' Sorting and Grouping feature to produce a report that sorts records by addresses.

A word about your address data

We'll first point out an important requirement your address data must meet before you can implement our technique: The street number and street name data must reside in separate fields. Most users combine this information into one field. If you want to split your single Address field into Street Number and Street Name fields, see "Parsing an Address Field to Derive Street Number and Street Name Fields," on page 5.

The technique

To sort address data for a delivery report, you must first group by the Street Name field and then group on the even and odd values in the Street Number field. If you've used Access' Sorting and Grouping feature before, you might already know how to sort by an ordinary field. You simply enter the field name—Street Name, in this case—in a row of the Sorting and Grouping window.

Next, to group by block, you use the Sorting and Grouping feature's capability to group by ranges of values. Since a new block begins every 100 street numbers, you group by the Street Number field in intervals of 100. You enter the field that stores the street number data in a row of the Sorting and Grouping window and, in the Group Properties section, set the Group On property to *Interval* and the Group Interval property to *100*.

Finally, sorting numeric data by even and odd values requires a more sophisticated technique. In a nutshell, you use Access' ability to sort records by the results of an expression by entering as the sorting expression *=[Street Number] Mod 2*. (We're assuming the name of the field, in this case, is Street Number.)

An example

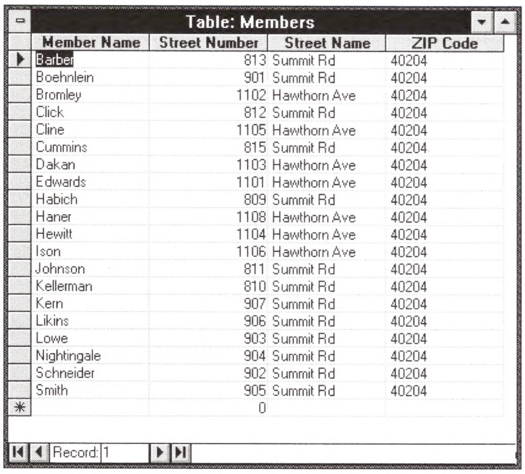
To demonstrate how to create a report that sorts by side of the street, let's suppose you

IN THIS ISSUE

manage the membership list of your neighborhood association. In preparing for your membership drive, you want to create a report that lists the members by the side of the street on which they live. That way, the volunteers who canvas the neighborhood to renew memberships can look at the report and anticipate the next house they need to visit.

Table A shows the structure of the Members table, which stores the names and addresses you want to list on the membership drive report. Figure A shows the sample data we'll use.

Figure A



Member Name	Street Number	Street Name	ZIP Code
Barber	813	Summit Rd	40204
Boehnlein	901	Summit Rd	40204
Bromley	1102	Hawthorn Ave	40204
Click	812	Summit Rd	40204
Cline	1105	Hawthorn Ave	40204
Cummins	815	Summit Rd	40204
Dakan	1103	Hawthorn Ave	40204
Edwards	1101	Hawthorn Ave	40204
Habich	809	Summit Rd	40204
Haner	1108	Hawthorn Ave	40204
Hewitt	1104	Hawthorn Ave	40204
Ison	1106	Hawthorn Ave	40204
Johnson	811	Summit Rd	40204
Kellerman	810	Summit Rd	40204
Kern	907	Summit Rd	40204
Likins	906	Summit Rd	40204
Lowe	903	Summit Rd	40204
Nightingale	904	Summit Rd	40204
Schneider	902	Summit Rd	40204
Smith	905	Summit Rd	40204
	0		

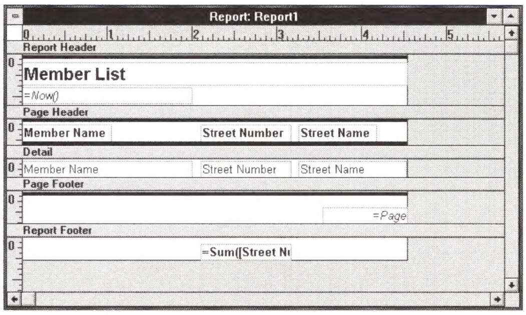
The report we'll show you lists these names and addresses by even and odd Street Number entries.

Table A

Key	Field Name	Data Type	Field Properties
🔑	Member Name	Text	Field Size=50
	Street Number	Number	Field Size=Integer
	Street Name	Text	Field Size=30
	ZIP Code	Text	Field Size=10

Now let's create the report. Return to the Database window, highlight the Members table, and click the New Report button (▤) on the tool bar. In the New Report dialog box, click the ReportWizards button. Then, in the dialog box that follows, highlight the Group/Totals option and click OK. Next, reply to the wizard's dialog boxes with the responses listed in Table B. When you click the Design button on the last dialog box, Access will generate the report shown in Figure B.

Figure B



The Group/Totals ReportWizard generates this report as the starting point for our membership drive report.

The ReportWizard creates one control you don't need on the report. You can remove the text box in the Report Footer that sums the numeric Street Number entries. (The Group/Totals wizard automatically sums numeric fields in the Report Footer.)

Inside MICROSOFT ACCESS™

Inside Microsoft Access (ISSN 1067-8204) is published monthly by The Cobb Group.

Prices:

Domestic	$59/yr. ($7.00 each)
Outside US	$79/yr. ($8.50 each)

Phone:

Toll free	(800) 223-8720
Local	(502) 491-1900
Customer Relations Fax	(502) 491-8050
Editorial Department Fax	(502) 491-4200

Address:
You may address tips, special requests, and other correspondence to

The Editor, *Inside Microsoft Access*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Postmaster:
Second class postage is pending in Louisville, KY. Send address changes to

Inside Microsoft Access
P.O. Box 35160
Louisville, KY 40232

Staff:

Editor-in-Chief	David Brown
Editing	Timothy E. Hampton Elizabeth Welch
Production Artist	Julie Jefferson
Design	Karl Feige
Publications Manager	Tara Dickerson
Managing Editor	Suzanne Thornberry
Circulation Manager	Brent Shean
Publications Director	Linda Baughman
Editorial Director	Jeff Yocom
Publishers	Mark Crane Jon Pyles

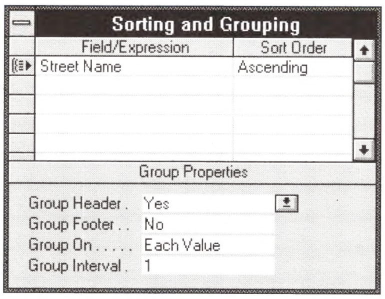
Back Issues:
To order back issues, call Customer Relations at (800) 223-8720. Back issues cost $7 each, $8.50 outside the US. You can pay with MasterCard, VISA, Discover, or American Express, or we can bill you. Please identify the issue you want by the month and year it was published. Customer Relations can also provide you with an issue-by-issue listing of all articles that have appeared in *Inside Microsoft Access*.

Table B

ReportWizard Question	Response
Which fields do you want on the report?	Member Name Street Number Street Name
Which fields do you want to group by?	None
Which fields do you want to sort by?	None
What kind of look do you want for your report?	Presentation
What title do you want for your report?	Member List

Next, we'll show you how to create the sort order for the report. Start by clicking the Sorting and Grouping button (▤) on the tool bar. In the first row, click the Field/Expression cell's dropdown arrow and select the Street Name field. Next, define a group header for this field by assigning *Yes* to the Group Header property in the Group Properties section, as shown in Figure C.

Figure C



Sorting and Grouping

Field/Expression	Sort Order
Street Name	Ascending

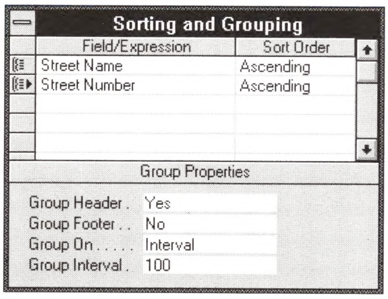
Group Properties

Group Header .	Yes
Group Footer . .	No
Group On	Each Value
Group Interval .	1

You group on the Street Name field to list the members by the streets they live on.

In the second row's Field/Expression cell, select Street Number. Then, move to the Group Properties section and set the Group Header property to *Yes*. Also, configure the row to group by intervals of 100 by setting the Group On property to *Interval* and the Group Interval property to *100*, as shown in Figure D.

Figure D



Sorting and Grouping

Field/Expression	Sort Order
Street Name	Ascending
Street Number	Ascending

Group Properties

Group Header .	Yes
Group Footer . .	No
Group On	Interval
Group Interval .	100

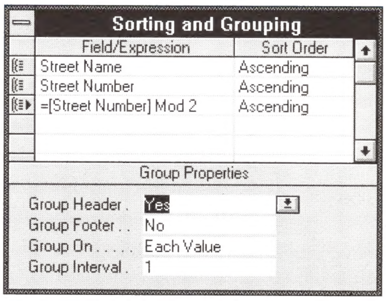
To further group members by block, group Street Number entries by intervals of 100.

Finally, in the third row's Field/Expression cell, type the expression

```
=[Street Number] Mod 2
```

After you move off the cell by pressing [Enter] or [Tab], Access will display the Group Properties section. Move to the Group Properties section by pressing [F6] and create a group header for this row by setting the Group Header property to *Yes*, as shown in Figure E.

Figure E



Sorting and Grouping

Field/Expression	Sort Order
Street Name	Ascending
Street Number	Ascending
=[Street Number] Mod 2	Ascending

Group Properties

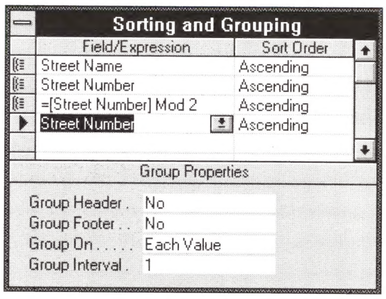
Group Header .	Yes
Group Footer . .	No
Group On	Each Value
Group Interval .	1

To group addresses by the side of the street the members live on, you group on an expression.

You aren't quite through yet. You must include one more row in the Sorting and Grouping window to define the sort order within the even and odd groups. In the fourth row's Field/Expression cell, click the dropdown arrow and select Street Number again from the selection list. You don't need to create any groups for this particular row. Figure F shows the completed Sorting and Grouping window.

Before continuing, save the report by pulling down the File menu and selecting the Save As... option. Enter *Members List For Canvasing* in the Save As dialog box and click OK.

Figure F



Sorting and Grouping

Field/Expression	Sort Order
Street Name	Ascending
Street Number	Ascending
=[Street Number] Mod 2	Ascending
Street Number	Ascending

Group Properties

Group Header .	No
Group Footer . .	No
Group On	Each Value
Group Interval .	1

You must define a sort field in order to sort the data within the groups.

Figure G, on the following page, shows the finished report along with the report output for the data we showed in Figure A. As you can see, the report first groups by Street Name entries, then by block, and finally by even and odd Street Number entries.

Fine-tuning the delivery-route report

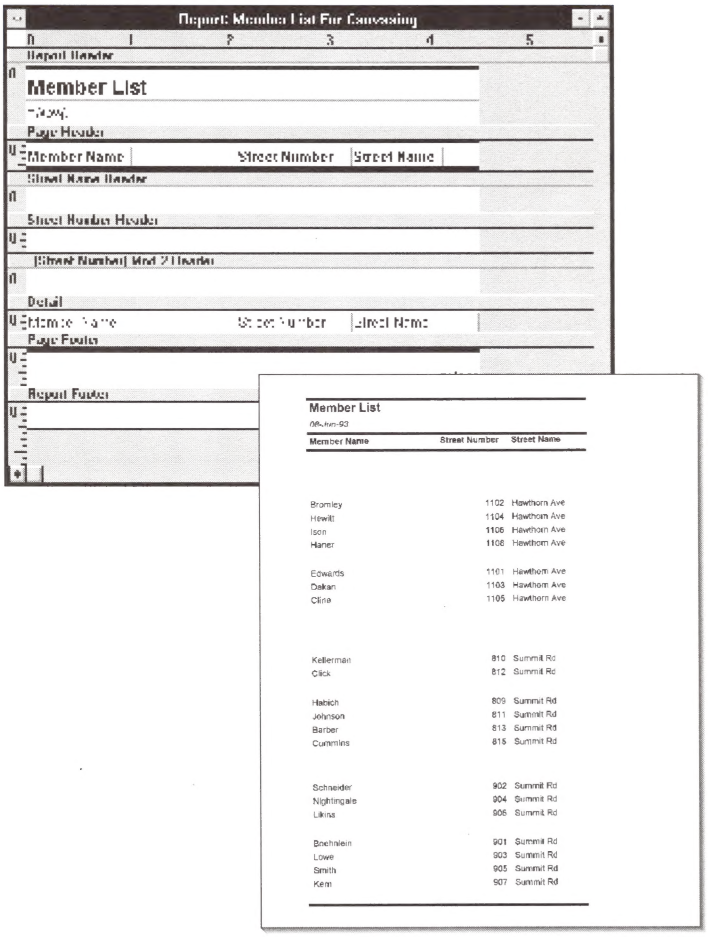
In most situations, your table will contain much more data than we've shown in our example. As a result, the report won't print the data on one neat page. When you print such a report for a real-world situation, you'll want to define page breaks for all your groups.

Furthermore, if you have quite a lot of data, you may want to give portions of the delivery-route report to several people. By defining the page breaks wisely, you can easily divide the report into streets and blocks so that each person can concentrate on a certain region.

To create a page break in a report, you open the property sheet for the group section in which you want the page breaks to appear. Then, set the Force New Page property to *After Section*. When you do so, Access will automatically move to a new page after printing each group of data.

For example, let's create the page break for the Street Name group. First, open the property sheet (if it isn't open already) by clicking the Properties button (▦) on the tool bar. Then, click the Street Name Header bar that identifies the Street Name Header section. The first property in the property sheet is Force New Page. Click this property's drop-down arrow and choose *After Section* from the selection list, as shown in Figure H.

Figure G



Report: Member List For Canvassing

Report Header

Member List

=Now()

Page Header

Member Name | Street Number | Street Name

Street Name Header

Street Number Header

=[Street Number] Mod 2 Header

Detail

Member Name | Street Number | Street Name

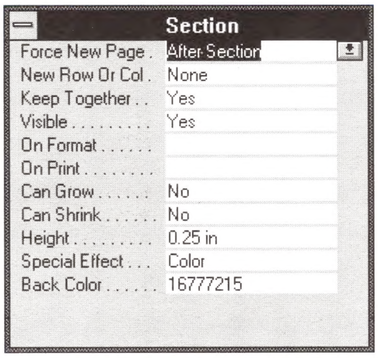
Page Footer

Report Footer

```
Member List
08-Jun-93
Member Name                    Street Number   Street Name


Bromley                              1102   Hawthorn Ave
Hewitt                               1104   Hawthorn Ave
Ison                                 1106   Hawthorn Ave
Haner                                1108   Hawthorn Ave

Edwards                              1101   Hawthorn Ave
Dakan                                1103   Hawthorn Ave
Cline                                1105   Hawthorn Ave



Kellerman                             810   Summit Rd
Click                                 812   Summit Rd

Habich                                809   Summit Rd
Johnson                               811   Summit Rd
Barber                                813   Summit Rd
Cummins                               815   Summit Rd


Schneider                             902   Summit Rd
Nightingale                           904   Summit Rd
Likins                                906   Summit Rd

Boehnlein                             901   Summit Rd
Lowe                                  903   Summit Rd
Smith                                 905   Summit Rd
Kern                                  907   Summit Rd
```

Here's the final report design along with its output.

Figure H



Section

Force New Page .	After Section
New Row Or Col .	None
Keep Together . .	Yes
Visible	Yes
On Format	
On Print	
Can Grow	No
Can Shrink	No
Height	0.25 in
Special Effect . . .	Color
Back Color	16777215

To create a page break between addresses on different streets, set the Street Name Header section's Force New Page to After Section.

Create a page break for the Street Number Header and the =[Street Number] Mod 2 Header sections as well. Remember, the Street Number Header section groups records by Street Number intervals of 100, and the =[Street Number] Mod 2 Header section groups the data by even and odd entries. Setting these page breaks will place records from each side of the street on each block on their own page. Then, you can divide those pages as appropriate for the task. For instance, you can split the delivery tasks on a long street among several people. Just give each person the pages that correspond to their assigned blocks.

Conclusion

In this article, we showed you how you can create a report that sorts address data for a delivery-route report. The report showcased several sorting and grouping techniques, including sorting by a range of values and by the results of an expression. ◆

Query Tip

Parsing an Address field to derive Street Number and Street Name fields

In "Creating Delivery-route Reports by Sorting Even and Odd Street Numbers," on page 1, we showed you how to group address data by block and by side of the street. However, you can't group the data this way if the components of the street address—the street number and street name—reside in one big Address field. To create a delivery-route report, you must group by these items individually.

In this article, we'll show you several Access Basic functions you'll need in order to manipulate the string entries in an Address field. Then, we'll show you a query technique that uses those functions to parse the components of the Address field into separate Street Number and Street Name fields.

The Val( ) function

The first function we'll describe is Val( ). This function accepts a string value as an argument and returns its numeric value. You often need the VAL( ) function when you store numeric data in a string format because you have to convert those string values to numbers before using them in calculations.

You might be wondering what the Val( ) function does when you provide a string that isn't a number. After all, the string *A* doesn't have a numeric value. Well, if the string you pass to the function doesn't begin with a number, the function returns 0. Also, if the string begins with a number but also contains letters and other characters, the function returns the numeric value of the number that begins the string.

Table A lists a few sample Val( ) calls that demonstrate how the function processes strings with both number and letter characters. The last example hints at how we'll use this function to parse the Address field. If we pass the field's entry—which almost always begins with the address' street number—the function will return the value we want to enter into the Street Number field.

The IIf( ) function

The next function we'll show you is the Immediate If function, IIf( ). This function evaluates a conditional expression and returns a particular value according to the result. You pass the expression in the first argument. In the second and third arguments, you pass the values you want the function to return when the expression evaluates to True and False, respectively.

You'll understand this function more clearly in an example. The function call

```
IIf(Number# = 0, "", Number#))
```

first evaluates the expression *Number# = 0*. If the number in the *Number#* variable equals 0, the expression evaluates to True, so the IIf( ) call returns a blank string. On the other hand, if *Number#* doesn't contain 0, the expression evaluates to False. Accordingly, the IIf( ) call will return the number.

The Right$( ), Len( ), and LTrim$( ) functions

Three other Access Basic functions help you extract the Street Name from the single Address field. We'll describe the Right$( ), Len( ), and LTrim$( ) functions and then explain how to use all the functions we've shown you in a query that parses street address data.

The Right$( ) function lets you extract a substring from the right side of a string. The function accepts two arguments—the string from which you want to extract the substring and the number of characters you want to extract. For instance, the function call

```
Right$("9420 Bunsen Parkway", 14)
```

returns the string *Bunsen Parkway*.

The Len( ) function accepts a single string argument and returns the number of characters in the argument. For example, the function call

```
Len("9420 Bunsen Parkway")
```

Table A

Expression	Return Value
`Val("34")`	34
`Val("P.O. Box 34")`	0
`Val("9420 Bunsen Parkway")`	9420

returns the number 19, since the address *9420 Bunsen Parkway* has 19 characters. As you'll see, you'll use this function to determine the number of characters the Right$( ) function should extract from the address string in order to return the street name string.

The LTrim$( ) function accepts a string argument and strips any leading blanks. For instance, the function call

```
LTrim$(" Bunsen Parkway")
```

returns the string *Bunsen Parkway*—without the argument's initial space character. If the string doesn't have leading blanks, the function simply returns the same string.

Parsing the Address field with a query

To demonstrate how to parse street address information, we'll use a slight variation in the Members table we defined in "Creating Delivery-route Reports by Sorting Even and Odd Street Numbers." Table B shows a new table named Members With Combined Addresses. Figure A shows some sample data.

Now let's build a query that parses the Address field into separate Street Number and Street Name fields. Start by highlighting the Members With Combined Addresses table in the Database window and clicking the New Query button (▣) on the tool bar. Next, drag the Member and ZIP Code fields to the QBE grid. Then, place the Member field in the first column and the ZIP Code field in the fourth column. (You'll type expressions into the second and third columns' Field cells.)

Figure A

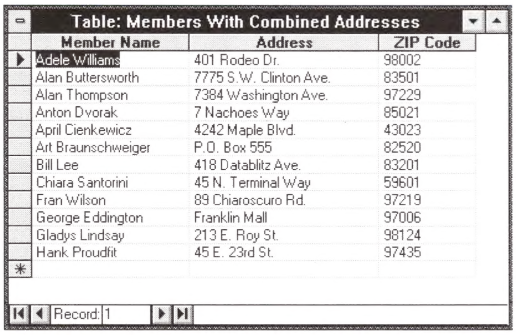


Table: Members With Combined Addresses

Member Name	Address	ZIP Code
Adele Williams	401 Rodeo Dr.	98002
Alan Buttersworth	7775 S.W. Clinton Ave.	83501
Alan Thompson	7384 Washington Ave.	97229
Anton Dvorak	7 Nachoes Way	85021
April Cienkewicz	4242 Maple Blvd.	43023
Art Braunschweiger	P.O. Box 555	82520
Bill Lee	418 Datablitz Ave.	83201
Chiara Santorini	45 N. Terminal Way	59601
Fran Wilson	89 Chiaroscuro Rd.	97219
George Eddington	Franklin Mall	97006
Gladys Lindsay	213 E. Roy St.	98124
Hank Proudfit	45 E. 23rd St.	97435

Record: 1

The query we'll show you parses the Address field into Street Number and Street Name fields.

Table B

Key	Field Name	Data Type	Field Properties
🔑	Member	Text	Field Size=50
	Address	Text	Field Size=50
	ZIP Code	Text	Field Size=10

Next, provide the expression that extracts the street number information. First, check the second column's Show box. Then, move the cursor to the Field cell and type

```
Street Number: IIf(Val([Address]) = 0, "",
➥  Val([Address]))
```

Notice you begin the Field cell entry with the *Street Number:* field label, which causes Access to name the field Street Number in the query datasheet.

The IIf( ) function evaluates the expression *Val([Address]) = 0*, which is true if the Address field begins with a number. If Val( ) returns 0, the field doesn't begin with a number, so IIf( ) returns an empty string to the Street Number field. If the Address field *does* begin with a number, IIf( ) returns that number to the Street Number field.

Next, enter into the third column the expression that extracts the Street Name. Again, check the third column's Show box and enter into the Field cell

```
Street Name: LTrim$(Right$([Address],
➥  Len([Address]) - Len([Street Number])))
```

Here, we label the new field Street Name. As you can see, the expression that extracts the street name information is more complicated than the previous expression.

We'll start our explanation by describing the role of the Right$( ) function, since it actually extracts the street name information from the full Address field entry. You specify the Address field as the string from which you want pull a substring. In the second argument, you must supply the number of characters the function should extract.

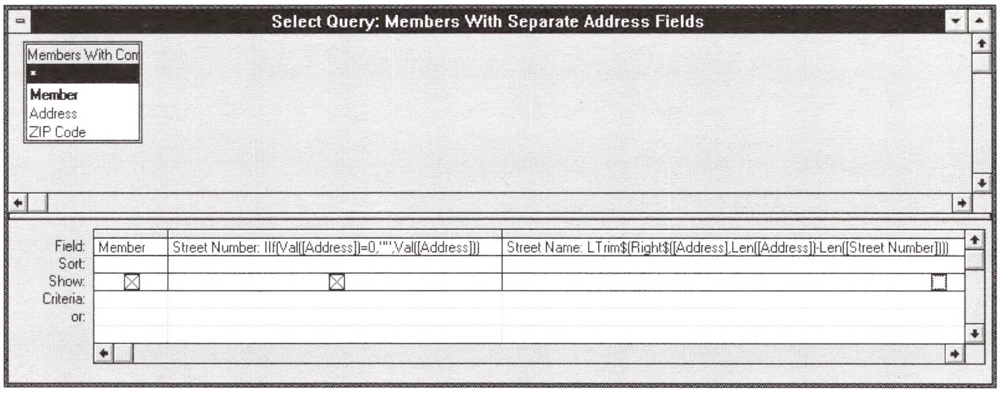
To determine the number of characters, the expression uses the Len( ) function to find the number of characters in both the Address field and the Street Number field. (Note that we use the Street Number field the query creates as it evaluates the expression in the grid's second column.) The difference between those two lengths is the number of characters that follows the street number.

When you use this number, the Right$( ) function returns the street name. However, it includes the space between the street number and street name as a leading space. Fortunately, the LTrim$( ) function makes stripping the leading space simple. By enclosing the Right$( ) function call in the LTrim$( )

function, you strip off this space, leaving only the street number. Then, the query includes the resulting value in the Street Number field.

Now that the query is complete, save it by pulling down the File menu and selecting the Save As... command. Enter *Members With Separate Address Fields* in the Save As dialog box and click OK.

Figure B



Select Query: Members With Separate Address Fields

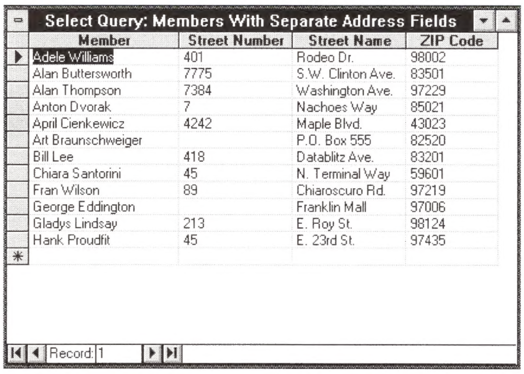
Members With Con
*
Member
Address
ZIP Code

Field:	Member	Street Number: IIf(Val([Address])=0,"",Val([Address]))	Street Name: LTrim$(Right$([Address],Len([Address])-Len([Street Number])))
Sort:			
Show:	☒	☒	□
Criteria:			
or:			

You create this query to parse the street address data in the Address field.

Figure B shows the new query. We've enlarged the Query window and the columns in the QBE grid in order to display the full expressions. Then, click the Datasheet View button (▦) on the tool bar. Access will produce the data-sheet shown in Figure C.

Figure C



Select Query: Members With Separate Address Fields

Member	Street Number	Street Name	ZIP Code
Adele Williams	401	Rodeo Dr.	98002
Alan Buttersworth	7775	S.W. Clinton Ave.	83501
Alan Thompson	7384	Washington Ave.	97229
Anton Dvorak	7	Nachoes Way	85021
April Cienkewicz	4242	Maple Blvd.	43023
Art Braunschweiger		P.O. Box 555	82520
Bill Lee	418	Datablitz Ave.	83201
Chiara Santorini	45	N. Terminal Way	59601
Fran Wilson	89	Chiaroscuro Rd.	97219
George Eddington		Franklin Mall	97006
Gladys Lindsay	213	E. Roy St.	98124
Hank Proudfit	45	E. 23rd St.	97435

Record: 1

The query creates this datasheet.

Basing the delivery-route report on the new query

Now that you've created a query that parses the street address data, you can create a report based on the query. You could create the report we describe in the article on page 1, which groups by block and side of the street. If you want to work through the article's example, simply highlight the query's name in the Database window and click the New Report button (▤) on the tool bar. Then, create the report as you normally would.

However, there's one important difference between creating the delivery-route report for this query and the Members table in the other article. The Street Number field the query creates is a text field. As a result, you won't be able to group on intervals of field entries or use the field in the expression that groups on even and odd values.

To create those types of groups when the Street Number field is a text value, you must use the Val( ) function to convert the entries to numbers, which you can then group on. For instance, if you want to group on intervals of 100, you enter the expression

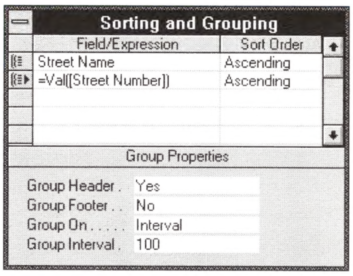
```
=Val([Street Number])
```

into the Sorting and Grouping window's Field/Expression cell instead of simply entering the field name. Then you set the Group On and Group Interval properties to define the grouping range. Figure D shows the result.

Conclusion

In this article, we showed you how to create a query that parses a single field and stores the street address information in separate Street Number and Street Name fields. You need such a query when you want to group the table's data by the Street Number or Street Name data. Once you've created a query that separates the components of the street address, you can base the report on the query that groups the data into delivery routes. ◆

Figure D



Sorting and Grouping

Field/Expression	Sort Order
Street Name	Ascending
=Val([Street Number])	Ascending

Group Properties

Group Header	Yes
Group Footer	No
Group On	Interval
Group Interval	100

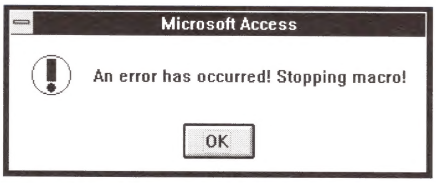
When Street Number is a text field, you must first convert the entries to numbers.

Macro Tip

Creating message boxes that offer choices during macros

If you've designed many macros, you've probably learned how useful the MsgBox action can be. This action pauses macro execution in order to tell the user about something that has happened. For instance, the macro could use the MsgBox action to display an error message, such as the one shown in Figure A, before quitting.

Figure A

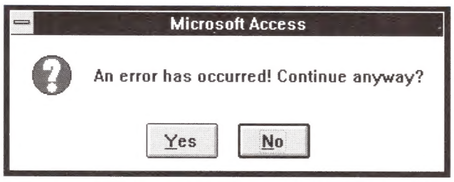


An error has occurred! Stopping macro!

The MsgBox macro action can produce informational message boxes that have only an OK button.

However, the MsgBox action has one major shortcoming—it lets you display only *informational* messages such as error messages. It does *not* create message boxes that give the user a choice of how to respond to the error condition. For example, the message box shown in Figure B gives you options when an error occurs. You can click the Yes button to continue with the macro despite the error or click the No button to stop the macro.

Figure B



An error has occurred! Continue anyway?

You often want to create message boxes that give your users choices.

In this article, we'll show you how to create *interactive* message boxes in macros. But, before we show you the technique in detail, we'll describe some of the difficulties you'll encounter while implementing this technique.

Use the MsgBox( ) function to create the message box

Creating a message box that asks a question is different from creating an ordinary message box: You must receive a user response as well as display a message. You must use the Access Basic function MsgBox( ) to create such a message box, since MsgBox( ) returns a value that indicates which button the user clicked.

The MsgBox( ) function is a lot like the MsgBox action. It creates a pop-up message box, offering you various message-box styles. However, MsgBox( ) also lets you choose from a variety of button arrangements. For instance, you can create a message box with Yes and No buttons. If the user clicks Yes, the function will return 6; if the user clicks No, it will return 7. For a detailed explanation of how to use the MsgBox( ) function, see "Understanding Access Basic's MsgBox( ) Function," on page 10.

Where you call MsgBox( )

The next issue you must address in implementing interactive message boxes is how to respond to the user's selection. Remember, you create an interactive message box so the macro can take different actions for the different possible user selections.

As you probably know, you ordinarily make decisions in macros by placing a conditional expression in the Condition cell. If the expression evaluates to True, the macro will execute that row's action.

The best way to create an interactive message box is by calling MsgBox( ) as part of the Condition cell's expression. To phrase the function call as a conditional expression, you test the function's return value.

A simple example

The technique may be clearer in the context of an example. Let's create a message box that asks users if they want to continue with the macro after an error occurred. The message box will display the message *An error occurred. Continue?* and will provide Yes and No buttons. If the user clicks No, you want the macro to execute the StopMacro action in order to abort the macro.

You create this message box by passing the message to the function's first argument. To include Yes and No buttons, you pass the number 4 to the function's second argument. Keep in mind that the MsgBox( ) function returns 7 when the user clicks the No button, so you want to execute the StopMacro action when the function returns 7.

Now, let's create a macro row that uses the MsgBox( ) function in the Condition column to create an interactive message box. First, create a new macro by moving to the Database window, clicking the Macro button to display the macros in the database, and then clicking New. When the Macro window appears, click the Conditions button () on the tool bar. Then, in the first row, enter the expression

```
MsgBox("An error occurred. Continue?",4) = 7
```

in the Condition cell. Next, move to the Action cell, click the dropdown arrow, and choose *StopMacro* from the selection list. Figure C shows the macro row. (We've expanded the Condition column to show the entire expression.)

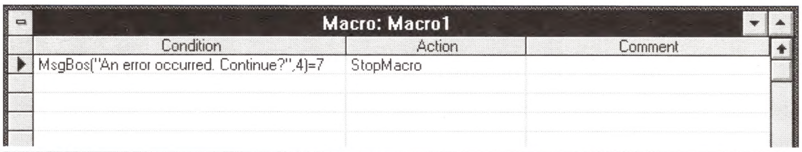
Let's add one more row to this simple macro so you can confirm this row's Condition cell reads your button click properly. We'll use the MsgBox action to create an informational message box that simply says the macro is completing normally.

Move to the second row's Action cell, click the dropdown arrow, and choose the MsgBox action from the selection list. Then, in the Action Arguments section, enter *The macro completed normally* in the Message argument.

Before running the macro, save it. Pull down the File menu and select the Save As... command. In the Save As dialog box, enter *MsgBox( ) Test* and click OK.

Now run the macro by clicking the Run button (!) on the tool bar. When Access evaluates the expression in the Condition cell, it will call the MsgBox( ) function, which will display the message box shown in Figure D. If the user clicks No, the function will return 7, and the expression in the Condition cell will evaluate to True. Then the macro will execute the StopMacro action. On the other hand, if the user clicks Yes, the macro will continue and execute the MsgBox macro statement on the second row, displaying the message box shown in Figure E.

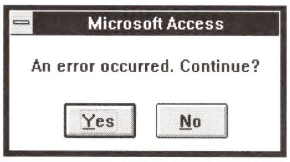
Figure C


Macro: Macro1

	Condition	Action	Comment
▶	MsgBos("An error occurred. Continue?",4)=7	StopMacro	

To create an interactive message box, you enter a MsgBox( ) statement in the Condition cell that determines which button the user clicks.

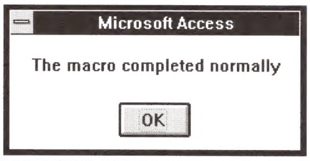
Figure D


Microsoft Access

An error occurred. Continue?

The MsgBox( ) function call creates this message box.

Figure E


Microsoft Access

The macro completed normally

If you click Yes in the previous figure's message box, the MsgBox action will display this message box.

Conclusion

In this article, we showed you how to use the MsgBox( ) function in a macro row's Condition cell in order to offer the user options during the macro. If you anticipate having the user make a decision during a macro, you can use this technique. Just ask a question by using the MsgBox( ) function in a Condition cell and respond to the answer with an action in the same row's Action cell. ◆

The [F2] key toggles the entry highlight in cells and controls

Here's a tip for data-entry users who find the mouse an often cumbersome way to operate a computer: The [F2] key toggles the cell highlight.

As you probably know, Access automatically highlights the field entries as you tab from cell to cell (or from control to control if you're using a form). As a result, if you immediately start typing, Access will replace the current entry. When you want only to modify the existing data, you must remove the highlight first.

If you'd rather not use the mouse to remove the highlight by clicking within the entry, you can simply press the [F2] key. When you do so, the insertion point cursor appears at the end of the entry. You can then use the arrow keys to move to characters you want to change.

Access Basic

Understanding Access Basic's MsgBox( ) function

The MsgBox( ) function will create several types of message boxes that offer various ways to communicate with the user. You can create a message box that has nothing more than an OK button, or you can create a message box with multiple buttons that give the user a choice of options. In this article, we'll tell you what you need to know to make the most of MsgBox( ).

Calling the MsgBox( ) function

You define the attributes of the message box by passing certain values in the MsgBox( ) function's arguments. The call to MsgBox( ) takes the form

```
n# = MsgBox(message,type,title)
```

having three arguments. In the first argument, *message*, you pass the message you want to display in the pop-up box. You pass to the *type* argument a numeric code that defines the message box style. We'll have much more to say about this code in a moment. Finally, you pass the window title you want in the message box to the *title* argument.

Also, the function returns a value that identifies the button the user pushed. We'll describe the possible codes later.

Table A

Code	Meaning
0	OK button only
1	OK and Cancel buttons
2	Abort, Retry, and Fail buttons
3	Yes, No, and Cancel buttons
4	Yes and No buttons
5	Retry and Cancel buttons

Table B

Code	Meaning	Icon
0	No icon	
16	Critical Message icon	🛑
32	Warning Query icon	?
48	Warning Message icon	!
64	Information Message icon	ℹ

Table C

Code	Meaning
0	First button is the default.
256	Second button is the default.
512	Third button is the default.

The *type* argument

As we mentioned, the *type* argument supplies all the information required to define the type of message box. With it, you specify the buttons that should appear, the default button, and the icon that resides next to the message.

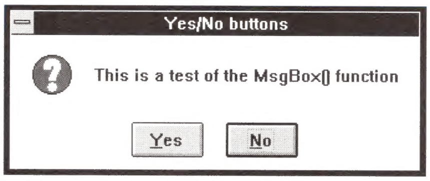
How does a single number do all this? Well, each attribute you define for the message box has a numeric code. You decide the attributes you want, then look up the corresponding codes and combine them into the single numeric value you use for the argument.

We'll show you how to combine the numeric codes shortly. Let's first look at the codes for the individual attributes. Table A lists the codes that specify the buttons in the message box, Table B lists the codes that specify the message box's icon, and Table C lists the codes that indicate which button will be the default.

When you compare the values in the tables, you'll notice that they begin with the code 0 but increment by different amounts: Table A's values increment by 1, Table B's values increase by 16, and Table C's values increment by 256. Access' developers structured the codes this way so you could simply add the values you choose from each table in order to create the final *type* argument.

To demonstrate how the individual codes combine, suppose you want to create a message box that has Yes and No buttons, displays the Warning Query icon (the question mark), and makes the No button—the second button in the message box—the default. You'd choose 4 from Table A, 32 from Table B, and 256 from Table C. You determine the *type* argument by computing 4+32+256, which equals 292. Figure A shows the type of message box this code defines.

Figure A



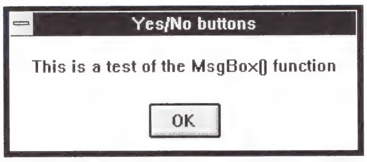
Yes/No buttons

This is a test of the MsgBox() function

The type *value 292 defines a message box that has Yes and No buttons—with No being the default button—and the Warning Query icon located next to the message.*

If you pass a number to the argument that doesn't represent a valid combination of these codes or if you don't even supply an argument, the function provides the default message box shown in Figure B, which has an OK button and no icon. Notice that using the code 0 also creates the default message box.

Figure B



This is a test of the MsgBox() function

The MsgBox( ) function creates the default message box when you pass 0 or an invalid code to the type *argument or you omit the* type *argument.*

The return code

When you create a message box with more than one button, you need to determine which button the user clicked. The MsgBox( ) function returns a code that tells you this information. Table D lists these return codes, which correspond to the types of buttons you can place on a message box. For instance, if the user clicks the Cancel button, the function will return 2.

Table D

Value	Button Clicked
1	OK
2	Cancel
3	Abort
4	Retry
5	Ignore
6	Yes
7	No

Conclusion

In this article, we described Access Basic's MsgBox( ) function and showed you how to create various types of interactive message boxes. See "Creating Message Boxes That Offer Choices During Macros," on page 8, for additional examples of using MsgBox( ). ◆

A new export option in Access 1.1 lets you create Word for Windows data files

Access Tip

Last month, we showed you how to use ordinary cut, copy, and paste commands to create the data files for your Word for Windows mail merge system ("Loading Access Data into a Word for Windows Data File"). As you may remember, the technique doesn't work very well when the names of the Access fields contain space characters. In this article, we'll show you an option that Access 1.1 provides to work around this problem.

Reviewing the Version 1.0 technique—and its problems

Let's first review the technique we showed you last month. In a nutshell, you can select records in an Access datasheet, copy them to the Clipboard, and then switch to Word for Windows and paste them into an empty Word for Windows document. When you paste the data into the word processor document, the field headers, which are necessary for Word for Windows data files, automatically appear.

Unfortunately, this technique has one problem. The field names appear exactly as they exist in the Access table. Consequently, if a name contains spaces, the spaces will transfer to the Word for Windows document. Since Word for Windows doesn't allow spaces in the names of a data file's field header, the paste operation won't produce a workable data file. You'll need to remove the spaces or substitute another character, such as the underscore, in the Word document's field header before attaching the header to a mail merge letter.

With the introduction of the Version 1.1 maintenance release, Access provides you with a new way to create Word for Windows data files: Access 1.1 offers a new option in the File menu's Export... command. It's now possible for you to export a table's or query's data to Word for Windows as well as to another Access database's table, a Paradox table, a FoxPro database, or any other of the many export options.

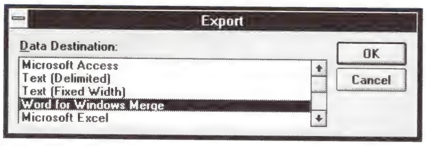
Exporting member data to a Word for Windows data file

Suppose you want to send a letter to your neighborhood association members. You use the table named Members shown in Table A on page 2 to keep track of member information, and you want to export the table's data to a Word for Windows document.

To do so, pull down the File menu while viewing the Database window and select the Export... option. This menu command kicks off a series of dialog boxes with which you provide the information Access requires to export the data.

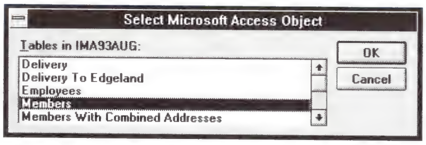
The Export dialog box is the first that appears. You'll see the new option, Word for Windows Merge, fourth on the list. (Access 1.1 introduces two other new options—FoxPro 2.0 and FoxPro 2.5.) You might recognize the others from the initial release. Highlight the Word for Windows Merge item, as shown in Figure A, and click OK.

Figure A



In order to export the data to a mail merge data file, select the Word for Windows Merge item in the Export dialog box.

Figure B



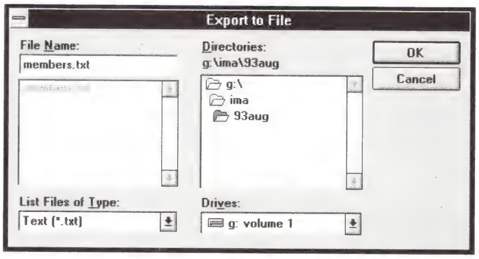
In the Select Microsoft Access Object dialog box, you select the table or query you want to export.

The next dialog box to appear is Select Microsoft Access Object. The selection list contains both the tables and the queries in the current database. Highlight the Members table, as shown in Figure B, and click OK.

Then, when the Export To File dialog box appears, you enter the filename you want to use for the data file. In this example, accept the default filename, MEMBERS.TXT, as shown in Figure C, and click OK. Be sure to note the directory selected in the Directories list so you'll know where to find the file later.

In the last dialog box, Export Word Merge Options, shown in Figure D, you can set several custom formats for certain Date/Time and Number data types. For example, you can access export dates in many formats, including the character Access uses to separate the month, day, and year. In this example, accept the default settings and simply select OK.

Figure C


File Name: members.txt

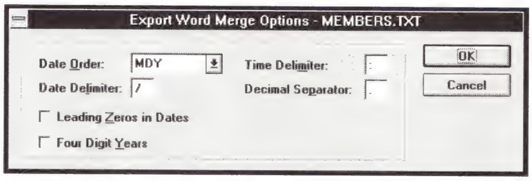
Directories: g:\ima\93aug

List Files of Type: Text (*.txt)

Drives: g: volume 1

You provide the filename of the data file in the Export To File dialog box.

Figure D


Date Order: MDY

Date Delimiter: /

Time Delimiter: :

Decimal Separator: .

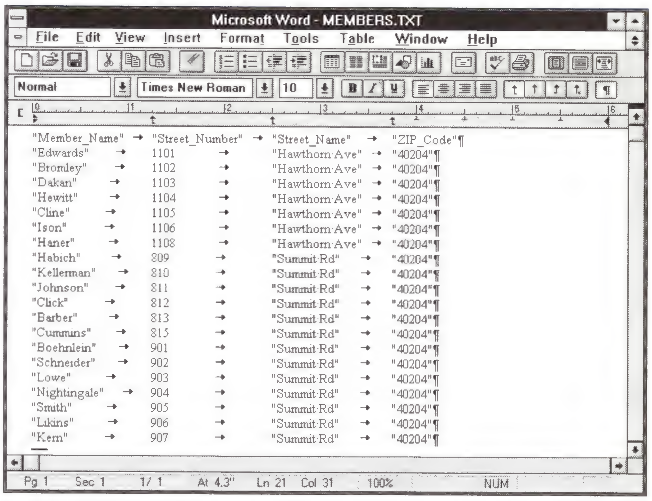
□ Leading Zeros in Dates

□ Four Digit Years

You can set various data-formatting options in the Export Word Merge Options dialog box.

Figure E shows the MEMBERS.TXT file, which Access creates to store the Members data in the Word for Windows data file. Notice the field header. Where the Access field names contain spaces, Access has substituted underscore characters. ◆

Figure E


```
"Member_Name"  →  "Street_Number"  →  "Street_Name"   →  "ZIP_Code"¶
"Edwards"      →  1101             →  "Hawthorn·Ave"  →  "40204"¶
"Bromley"      →  1102             →  "Hawthorn·Ave"  →  "40204"¶
"Dakan"        →  1103             →  "Hawthorn·Ave"  →  "40204"¶
"Hewitt"       →  1104             →  "Hawthorn·Ave"  →  "40204"¶
"Cline"        →  1105             →  "Hawthorn·Ave"  →  "40204"¶
"Ison"         →  1106             →  "Hawthorn·Ave"  →  "40204"¶
"Haner"        →  1108             →  "Hawthorn·Ave"  →  "40204"¶
"Habich"       →  809              →  "Summit·Rd"     →  "40204"¶
"Kellerman"    →  810              →  "Summit·Rd"     →  "40204"¶
"Johnson"      →  811              →  "Summit·Rd"     →  "40204"¶
"Click"        →  812              →  "Summit·Rd"     →  "40204"¶
"Barber"       →  813              →  "Summit·Rd"     →  "40204"¶
"Cummins"      →  815              →  "Summit·Rd"     →  "40204"¶
"Boehnlein"    →  901              →  "Summit·Rd"     →  "40204"¶
"Schneider"    →  902              →  "Summit·Rd"     →  "40204"¶
"Lowe"         →  903              →  "Summit·Rd"     →  "40204"¶
"Nightingale"  →  904              →  "Summit·Rd"     →  "40204"¶
"Smith"        →  905              →  "Summit·Rd"     →  "40204"¶
"Likins"       →  906              →  "Summit·Rd"     →  "40204"¶
"Kern"         →  907              →  "Summit·Rd"     →  "40204"¶
```

The Export feature writes the data in a mail merge data file format, including the field header.

Macro Tip

The Confirm Delete macro lets you guard against accidental record deletions

The article "Creating Message Boxes That Offer Choices During Macros," on page 8, describes a technique that lets you create interactive message boxes that allow you to make decisions during macro execution. In this article, we'll show you how to use that technique to create a macro that adds extra protection against accidental record deletion during data-entry sessions.

Why you may want extra protection

Access already produces a warning dialog box when you delete a record. The dialog box in Figure A appears, telling you a record has been deleted and giving you a chance to abort your deletion. If you click the Cancel button, Access will restore the record to the table. If you click OK, Access will permanently delete the record, and you won't be able to use the Undo command to restore it.

The protection feature's most distinguishing characteristic is the dialog box, which appears *after* you delete the record. When you click Cancel, you only undo the deletion. In some situations, you might want to add a dialog box that lets you abort the record deletion *before* the record disappears. For instance, you rarely want to delete records from a table that holds employee information. Even if you fire someone, you usually don't just delete the person's record of employment. With an extra message box warning you about an impending record deletion, you can immediately cancel any accidental deletions.

The technique

Now we'll describe how to incorporate such a message box in your data-entry forms. First, you create a macro that pops up a message box asking you whether you want to delete the record and cancels if necessary. Then, you assign this macro to the On Delete property of the form for which you're providing this protection. After you do so, Access will run the macro and pop up the new message box whenever you delete a record using the form.

Let's create the macro. Move to the Database window by pressing [F11], click the Macro button to show the list of macros in the database, and then click the window's New button. When the Macro window appears, click the Conditions button on the tool bar.

The macro we'll show you includes a single row. In the Condition cell, enter the expression

```
MsgBox("Are you sure you want to delete?",
➥  4+32+256, "Attention!") = 7
```

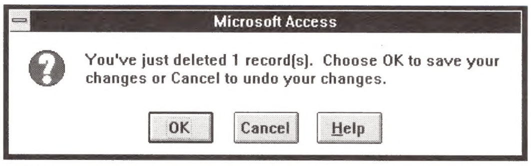
which displays the message box and tests whether the function returns the value 7. (MsgBox( ) returns 7 when you click the message box's No button.)

Let's take a moment to examine the function call's arguments. The first argument supplies the message the message box will display—*Are you sure you want to delete?*. The second argument defines the type of message box. As you can see, we express the argument as the sum of the codes that define the type of message box. That is, 4 specifies the Yes and No buttons, 32 selects Warning Query (?) as the icon to appear in the message box, and 256 selects the second button (No, in this case) as the default button. Finally, the third argument provides the window title of the message box—*Attention*.

Now provide the action for the macro row. Click the Action cell's dropdown arrow and then select CancelEvent. Figure B shows the completed action. (We've expanded the Condition column to show the entire expression.)

Finally, save the new macro by pulling down the File menu and selecting the Save As... option.

Figure A

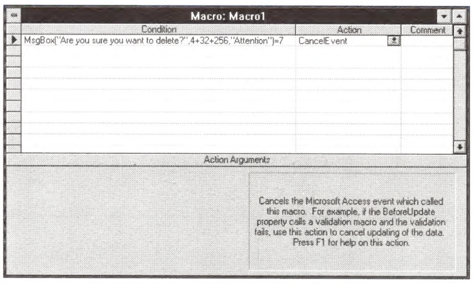


Microsoft Access

You've just deleted 1 record(s). Choose OK to save your changes or Cancel to undo your changes.

Access will display this dialog box when you delete a record.

Figure B



Macro: Macro1

Condition	Action	Comment
MsgBox("Are you sure you want to delete?",4+32+256,"Attention")=7	CancelEvent	

Action Arguments

Cancels the Microsoft Access event which called this macro. For example, if the BeforeUpdate property calls a validation macro and the validation fails, use this action to cancel updating of the data. Press F1 for help on this action.

If you assign this macro to a form's On Delete property, the macro will ask you for confirmation when you delete a record.

In the Save As dialog box, enter *Confirm Delete* and click OK. Then, close the Macro window.

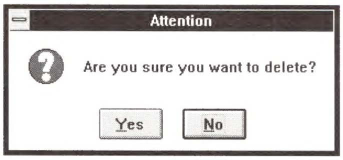
Understanding the Confirm Delete macro

Let's examine how the macro works during a data-entry session. Remember, you assign this macro to a form's On Delete property so that when you delete a record using the form, Access will run the macro before actually deleting the record.

The macro will then pop up the message box shown in Figure C as it evaluates the Condition cell. If you click Yes to confirm the record deletion, the macro won't execute the action; it will just stop. Next, Access will delete the record normally, producing a confirmation dialog box that lets you know the record was deleted.

If you click No in the Confirm Delete macro's message box, the macro will execute the CancelEvent action, which suppresses the action you used to delete the record. You may have used the Delete command on the Edit menu or pressed [Del].

Figure C



Attention

Are you sure you want to delete?

The Confirm Delete macro pops up this message box.

Figure D

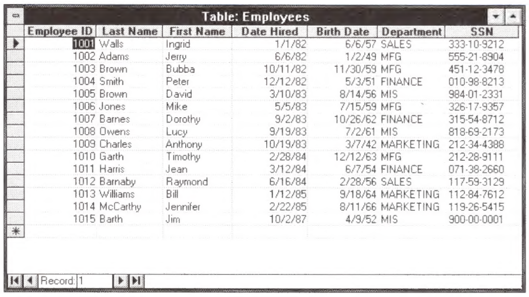


Table: Employees

Employee ID	Last Name	First Name	Date Hired	Birth Date	Department	SSN
1001	Walls	Ingrid	1/1/82	6/6/57	SALES	333-10-9212
1002	Adams	Jerry	6/6/82	1/2/49	MFG	555-21-8904
1003	Brown	Bubba	10/11/82	11/30/59	MFG	451-12-3478
1004	Smith	Peter	12/12/82	5/3/51	FINANCE	010-98-8213
1005	Brown	David	3/10/83	8/14/56	MIS	984-01-2331
1006	Jones	Mike	5/5/83	7/15/59	MFG	326-17-9357
1007	Barnes	Dorothy	9/2/83	10/26/62	FINANCE	315-54-8712
1008	Owens	Lucy	9/19/83	7/2/61	MIS	818-69-2173
1009	Charles	Anthony	10/19/83	3/7/42	MARKETING	212-34-4388
1010	Garth	Timothy	2/28/84	12/12/63	MFG	212-28-9111
1011	Harris	Jean	3/12/84	6/7/54	FINANCE	071-38-2660
1012	Barnaby	Raymond	6/16/84	2/28/56	SALES	117-59-3129
1013	Williams	Bill	1/12/85	9/18/64	MARKETING	112-84-7612
1014	McCarthy	Jennifer	2/22/85	8/11/66	MARKETING	119-26-5415
1015	Barth	Jim	10/2/87	4/9/52	MIS	900-00-0001

The Employees table stores name and address information.

Using the Confirm Delete macro in a form

Let's create a simple form for a table named Employees, which you'll use for testing the Confirm Delete macro. Table A shows the structure of the Employees table, and Figure D shows some sample data.

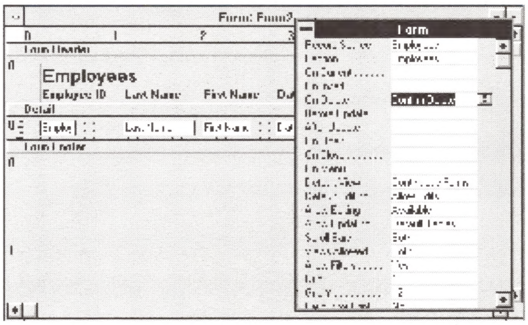
Table A

Key	Field Name	Data Type	Field Properties
🔑	Employee ID	Number	Field Size=Integer
	Last Name	Text	Field Size=15
	First Name	Text	Field Size=10
	Date Hired	Date/Time	
	Birth Date	Date/Time	
	Department	Text	Field Size=10
	SSN	Text	Field Size=11

Now create the form. In the Database window, click the Table button and highlight the Employees table. Then, click the New Form button (▤) on the tool bar. In the New Form dialog box, click the FormWizards button. In the following dialog box, select Tabular as the AccessWizard you want and click OK. Next, click the FastForward button (>>|) in the dialog box that follows to accept all default options to the wizard's questions. In the next dialog box, click the Design button.

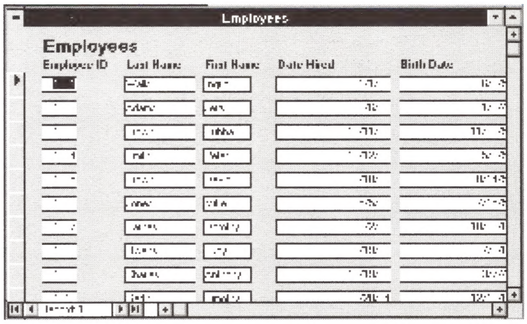
When the new form appears in Design View, click the Properties button (▤) on the tool bar to open the Property Sheet. Then, move to the On Delete property, click the dropdown arrow, and select Confirm Delete from the list of macros. Figure E shows the form at this point. To test the Confirm Delete macro's operation, click the Form View button (▤) on the tool bar to display the form, as shown in Figure F.

Figure E



You assign the Confirm Delete macro to the On Delete property by using the Property Sheet.

Figure F



The form lists the records in the table.

Next, click a record's record selector and press [Del]. The message box shown in Figure C appears. If you click No, the macro will cancel your deletion. If you click Yes, the macro will allow your deletion to proceed. The record will disappear, and Access will display the normal dialog box, which gives you yet another chance to cancel. ◆

What enhancements does Access Version 1.1 bring?

In early June, Microsoft began shipping the Access maintenance release—Version 1.1. This article will list the major enhancements and new features that the update provides. The new version costs $14.95 plus shipping and handling. You need only one copy to upgrade all the copies of Access 1.0 you've registered for your company.

Changes to the database file format

Access 1.1 enhances some properties of the database file. For instance, the maximum file size is 1 Gb in Version 1.1—an increase from 128 Mb. Also, the maintenance release replaces the Nordic sort order with three separate database sort orders—Swedish/Finnish, Norwegian/Danish, and Icelandic.

Note that to take advantage of these improvements, you must convert your existing databases to the new file format. Converting a database is easy: Before you open a database, you simply issue the Compact Database... command that resides on the File menu. However, if you convert a database, you must upgrade to Version 1.1 all users who work with that database. Access 1.0 is incompatible with the new file format.

Connectivity enhancements

If you hope to use Access with other data formats such as SQL data or FoxPro, you'll probably be very happy with Access 1.1. Table A describes the many data connectivity improvements in Access 1.1.

Miscellaneous improvements

Access 1.1 contains other important enhancements. Although we won't describe them in detail here, look for articles that explore those new features in future issues.

If you work on a network, you'll be happy to hear that Access 1.1 provides extra security safeguards. As you may know, Version 1.0 allows unscrupulous users to copy a database owner's SYSTEM.MDA file. As a result, the user becomes a member of the database's Admin group, which gives the user absolute control over the data and objects in the database. Under these circumstances, the intruder can even kick the true administrators out of the Admin group! In Access 1.1, you can ensure the SYSTEM.MDA remains unique.

Also, the new version makes creating and installing custom wizards much easier. You can create your own utility databases and "install" them as wizards. You can then invoke the utility databases as easily as the form and report wizards that come with Access. Also, you should watch for third-party products coming on the market. Developers across the country will be selling utility wizards you can buy to simplify your work with Access. ◆

Table A

Connectivity Enhancement	Comments
Exporting to Word for Windows mail merge data files	See "A New Export Option in Access 1.1 Lets You Create Word for Windows Data Files," on page 11, for a complete description of this new feature.
Seamless connectivity to FoxPro	You can import and export FoxPro databases and also attach them to an Access database to share the data with existing FoxPro applications.
New ODBC drivers	You can now attach tables from the ORACLE and Sybase's SQL Server database managers. Also, Microsoft has improved the efficiency of the existing ODBC driver for Microsoft's SQL Server database manager.
Enhanced importing facility for fixed-length text files	Access 1.1's import option for fixed-length text files isn't as rigid as Version 1.0's. Namely, it doesn't care if the last field in the record consumed the entire allowable field width. Access 1.0 would consider records that didn't use all the allotted space as variable in length and wouldn't import those records.
Importing and exporting to Excel spreadsheets' Database named range	When you transfer data to and from Excel spreadsheets, you can communicate directly with the special named range called Database.
Better connectivity to BTrieve tables	Version 1.1 simplifies importing and exporting BTrieve data as well as attaching that data. The enhancements don't necessarily improve how Access manages BTrieve data. Only configuring Access' connection to the data is better.
Better control over login IDs and passwords for SQL databases	Version 1.1 provides you with the ability to prevent users from storing their passwords locally.
Importing and attaching data on read-only drives	You can now import and attach data in dBASE, FoxPro, and BTrieve tables on read-only drives such as CD-ROM drives. You can't access Paradox data, since Access supports only the Paradox 3.x file format and Paradox didn't provide access to read-only drives until Version 4.0.
Importing and attaching external tables while they're in use	In Version 1.0, Access required exclusive use of the external table or database before importing the data or attaching it to your Access database. Access 1.1 eliminates that restriction.

SECOND CLASS MAIL

Microsoft Access
Technical Support
(206) 635-7050

Please include account number from label with any correspondence.

Letters

Why Access sometimes prints a blank page between your report's pages

I have a problem with Access reports that you may be able to help me with. I've created a report for my inventory database that always prints a blank page after every printed page. I've set the Force New Page properties of all the report sections to No. However, those properties don't seem to be the problem here. Do you have any suggestions?

Jon Olbum
Pittsburgh, Pennsylvania

A lot of users have asked Mr. Olbum's question. The problem arises because the width of the report—plus the left and right margins—is larger than the width of the actual piece of paper in the printer. Access prints an extra page for the overflow.

This fact isn't always obvious, since the rightmost edge of the report is an empty margin. If the total report width just barely exceeds the page size, Access will cause the margin to overflow onto the next page. The overflow isn't visible and the extra blank page can be a little mystifying.

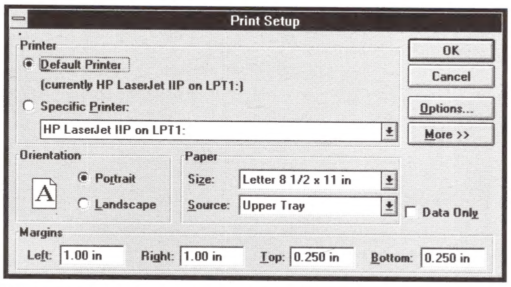
However, now that you know why Access prints the extra page, this behavior can serve as a useful troubleshooting tool. When you see the extra page, you'll know there isn't enough room on the page for your report and margin definitions. You can then find a way to reduce the report's width. If you can't do that, you may be able to adjust the margins in the Print Setup dialog box.

Remember, the total report width—the report width in the report's Design View window plus the left and right margins—must be equal to or less than the paper size. For instance, if you print on standard 8 ½ x 11-inch paper in the portrait configuration, the total report width must be equal to or less than 8 ½ inches.

If you want to reduce the margins, you do so in the Print Setup dialog box, shown in Figure A. You set the values in the Margin section at the bottom of the window in the Left and Right text boxes.

To display this dialog box during Design view, pull down the File menu and select the Print Setup... command. If you're previewing the report, you can view the Print Setup dialog box by clicking the Setup... button on the tool bar. ◆

Figure A



Printer
● Default Printer
(currently HP LaserJet IIP on LPT1:)
○ Specific Printer:
HP LaserJet IIP on LPT1:

Orientation
● Portrait
○ Landscape

Paper
Size: Letter 8 1/2 x 11 in
Source: Upper Tray
□ Data Only

Margins
Left: 1.00 in
Right: 1.00 in
Top: 0.250 in
Bottom: 0.250 in

You define the left and right margins in the Print Setup dialog box.

Subscribe to the Inside Microsoft Access Resource Disk!

Do you like the forms, reports, tables, macros, modules, and queries we regularly feature in *Inside Microsoft Access* but don't have the time or patience to create them? If so, then you may want to subscribe to the Inside Microsoft Access Resource Disk, which will be available next month. Once you subscribe, we'll send you a monthly disk loaded with all the useful tips featured in *Inside Microsoft Access*.

A six-month subscription to the Inside Microsoft Access Resource Disk costs $29. A full one-year subscription is $49. If you don't want to subscribe but would like the forms, reports, tables, macros, modules, and queries in a particular issue of *Inside Microsoft Access*, you can purchase the disk for only $9.95.

To subscribe or order a specific month's disk, just call our Customer Relations department at (800) 223-8720. Outside the US, please call (502) 491-1900.

Printed in the USA
Printed on recycled and recyclable paper
50% recycled fiber including 10% post-consumer waste